KOMPRESI FILE WAVE DENGAN ALGORITMA HUFFMAN

Nirsal
Universitas Cokroaminoto Palopo

Abstrak

Penelitian ini bertujuan untuk (1) untuk mengetahui cara kerja dari algoritma Huffman yang dipakai dalam kompresi dan dekompresi *file Wave*, (2) untuk menghasilkan sebuah perangkat lunak yang dapat melakukan kompresi dan dekompresi pada *file Wave* dengan input berupa sebuah *file Wave* serta sebagai *player file Wave*. Untuk menyelesaikan masalah yang ada, terdapat beberapa tahapan yang harus dilalui yaitu: (1) melakukan pengumpulan berbagai data dan informasi yang berkaitan dengan struktur *file Wave* dan algoritma Huffman untuk mendukung perangkat lunak yang akan dirancang penulis, (2) merancang antarmuka pemakai (*user interface*), (3) langkah penyelesaian program dimulai dari membaca *file Wave* untuk mengambil informasi dari *file* tersebut, mengambil *chunk data* pada *file Wave*, melakukan kompresi pada *chunk data* tersebut dan terakhir menulis kembali hasil data terkompresi tersebut beserta informasi *file Wave* tersebut ke dalam bentuk *file Wave* tersebut. (4) Menulis kode program dalam bahasa Visual Basic (5) Melakukan berbagai pengujian pada perangkat lunak yang dirancang dan memperbaiki kesalahan yang terdapat dalam aplikasi. Hasil penelitian menunjukkan bahwa (1) reduksi ukuran *file* yang diperoleh dengan algoritma Huffman ini berkisar dari *range* 20% hingga 40%. (2) tingkat kompresi dipengaruhi oleh banyaknya nada yang sama dalam *file Wave*, (3) File Wave yang telah dikompresi bila dilakukan proses kompresi sekali lagi maka ukuran *file* akan bertambah besar sedikit karena algoritma Huffman merupakan *optimal compression* jadi *file* yang dilakukan kompresi sebanyak dua kali maka proses terakhir tidak akan mereduksi ukuran *file* lagi.

Kata kunci: *Kompresi, Wave dan algoritma Huffman*

I. PENDAHULUAN

1.1 Latar Belakang

Salah satu *file* format suara yang banyak dipakai dalam sistem operasi Windows adalah format Wave (*.WAV). Format ini banyak digunakan untuk keperluan *game* dan *multimedia*. Wave sebenarnya merupakan format kasar (*raw format*) dimana signal suara langsung direkam dan dikuantisasi menjadi data digital. Format dasar dari *file* ini secara default tidak mendukung kompresi dan dikenal dengan nama PCM (Pulse Code Modulation).

Jika direkam suatu lagu sekualitas CD Audio menggunakan *sampling rate* 44,1 kHz, 16 *bit per sample*, 2 kanal (stereo), maka total media yang diperlukan untuk menyimpan data audio ini per detik adalah 176.400 *byte* sehingga untuk durasi 1 menit diperlukan 10,584 MB. Jika rata-rata durasi satu lagu selama 5 menit, maka dibutuhkan tempat lebih dari 50 MB untuk menyimpan data audio lagu tersebut. Ini tentunya sangat memboroskan media penyimpanan seperti hard disk meskipun saat ini telah tersedia kapasitas hard disk yang besar. Masalah tersebut dapat diatasi bila *file* Wave tersebut dikompresi untuk mengurangi ukurannya.

Beranjak dari masalah ini, maka akan dibuat sebuah perangkat lunak yang dapat melakukan kompresi pada *file* Wave sekaligus mampu memainkan kembali *file* Wave terkompresi tersebut. Maka dalam penelitian ini penulis mengambil judul "**Kompresi File Wave Dengan Algoritma Huffman**". Sesuai dengan latar belakang pemilihan judul, maka yang menjadi masalah dalam penelitian ini adalah merancang suatu perangkat lunak yang dapat melakukan kompresi pada *file* Wave dengan algoritma Huffman dan bagaimana cara memainkan kembali *file Wave* yang telah terkompresi tersebut. Agar pembahasan tidak menyimpang dari tujuan dilakukan pembatasan masalah sebagai berikut:

1. *File* Input hanya berupa *file* format *Wave*.
2. Program tidak dapat melakukan pengubahan jumlah kanal (*channel*), *bit*

per sample, dan *sampling rate file Wave*
3. Program dapat memainkan kembali *file Wave* terkompresi tersebut dengan pilihan *Play*, *Stop*, dan *Pause*.
4. Perancangan dan pembuatan perangkat lunak ini menggunakan bahasa Microsoft Visual Basic 6.0

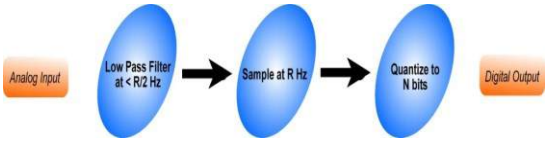
II. LANDASAN TEORI

2.1 Pengertian Audio Digital

Suara yang kita dengar sehari-hari adalah merupakan gelombang *analog*. Gelombang ini berasal dari tekanan udara yang ada di sekeliling kita, yang dapat kita dengar dengan bantuan gendang telinga. Gendang telinga ini bergetar, dan getaran ini dikirim dan diterjemahkan menjadi informasi suara yang dikirimkan ke otak, sehingga kita dapat mendengarkan suara. Suara yang kita hasilkan sewaktu berbicara berbentuk tekanan suara yang dihasilkan oleh pita suara. Pita suara ini akan bergetar, dan getaran ini menyebabkan perubahan tekanan udara, sehingga kita dapat mengeluarkan suara.

Proses pengubahan dari tegangan *analog* ke data *digital* ini terdiri atas beberapa tahap yang ditunjukkan pada Gambar 2.1, yaitu:

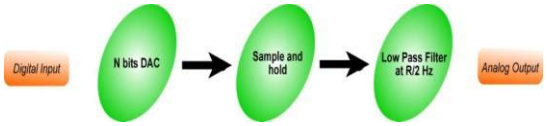
1. Membatasi frekuensi sinyal yang akan diproses dengan *Low Pass Filter*.
2. Mencuplik sinyal *analog* ini (melakukan *sampling*) menjadi beberapa potongan waktu.
3. Cuplikan-cuplikan ini diberi nilai eksak, dan nilai ini diberikan dalam bentuk data *digital*.



Gambar 1. Konversi Sinyal Analog ke Digital

Proses sebaliknya, yaitu pengubahan dari data *digital* menjadi tegangan *analog* juga terdiri atas beberapa tahap, yang ditunjukkan pada gambar 2.2, yaitu:

1. Menghitung data *digital* menjadi amplitudo-amplitudo *analog*.
2. Menyambung amplitudo *analog* ini menjadi sinyal *analog*.
3. Memfilter keluaran dengan *Low Pass Filter* sehingga bentuk gelombang keluaran menjadi lebih mulus.



Gambar 2. Konversi Sinyal Digital ke Analog

Proses pengubahan sinyal *analog* menjadi *digital* harus memenuhi sebuah kriteria, yaitu kriteria Nyquist. Kriteria ini mengatakan bahwa untuk mencuplik sebuah sinyal yang memiliki frekuensi X Hertz, maka harus mencupliknya minimal dua kali lebih rapat, atau 2X Hertz. Jika tidak, sinyal tidak akan dapat dikembalikan ke dalam bentuk semula.

2.2 Algoritma Kompresi Huffman

Algoritma kompresi Huffman dinamakan sesuai dengan nama penemunya yaitu David Huffman, seorang profesor di MIT (*Massachusets Instuate of Technology*).

Kompresi Huffman merupakan algoritma kompresi *lossless* dan ideal untuk mengkompresi teks atau *file* program. Ini yang menyebabkan mengapa algoritma ini banyak dipakai dalam program kompresi.

Contoh praktis berikut ini menunjukkan cara kerja dari algoritma Huffman. Misalkan akan dikompresi potongan data seperti berikut ini:

ACDABA

Distribusi frekuensi untuk karakter di atas seperti berikut ini:

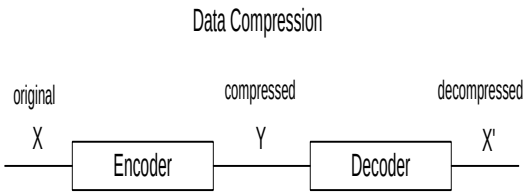
Karakter	A	B	C	D
Frekuensi	3	1	1	1

III. PEMBAHASAN DAN PERANCANGAN

3.1 Pembahasan

Kompresi data atau dikenal juga sebagai pemadatan data mempunyai tujuan memperkecil ukuran data sehingga selain dapat menghemat media penyimpanan dan memudahkan transfer dalam jaringan seperti Internet misalnya.

Contoh standar yang menggunakan jenis ini adalah Gzip, Unix Compress, WinZip, GIF (*Graphic Interchange Forma*t) untuk *still image*, dan Morse Code.



Gambar 3. Skema Proses Kompresi dan Dekompresi

X, Y, X' = String
Lossless Compression: X = X'
Lossy Compression: X != X'
Compression Ratio: |X| / |Y| dimana |X| adalah jumlah *bit* dalam dalam X dan |Y| adalah jumlah *bit* dalam Y

1. Encoding Huffman

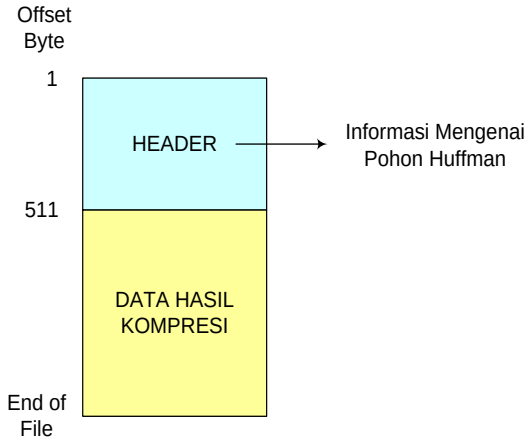
Algoritma kompresi Huffman atau disebut dengan *encoding* Huffman adalah algoritma yang dipakai untuk mengkompresi *file*. Teknik kompresi ini dengan menggantikan code yang lebih kecil pada karakter yang sering dipakai dan *code* yang lebih panjang untuk karakter yang tidak begitu sering dipakai.

Code dalam hal ini adalah urutan bit berupa nilai '0' dan '1' yang secara unik merepresentasikan sebuah karakter.

Ide dasar dari *encoding* Huffman adalah mencocokkan *code word* yang pendek pada blok input dengan kemungkinan yang terbesar dan *code word* yang panjang dengan kemungkinan terkecil. Konsep ini mirip dengan *Morse Code*.

2. Struktur File Hasil Kompresi

Hasil kompresi pada *file Wave* akan mempunyai ekstensi *.cmp dan dibentuk dengan struktur yang sederhana. Terdiri atas dua bagian yaitu bagian "Header" dan bagian "Data". Bagian "Header" merupakan bagian awal data yang berisi in*forma*si mengenai pohon Huffman pada *file Wave* yang dikompresi. Bagian ini mempunyai ukuran maksimum 511 *byte* dan bervariasi sesuai dengan banyaknya karakter yang terdapat pada *file Wave*. "Header" merupakan bagian yang penting untuk proses dekompresi nantinya untuk membentuk kembali pohon Huffman *file Wave* tersebut. Sedangkan bagian "Data" merupakan data hasil kompresi atas *file Wave* berdasarkan tabel kode dari pohon Huffman yang dihasilkan.



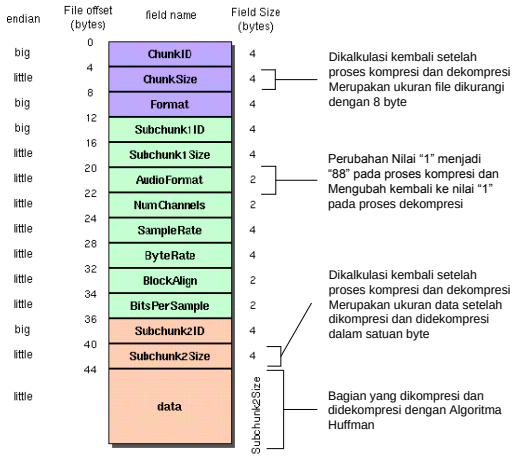
Gambar 4. Struktur File Hasil Kompresi

3.2 Perancangan

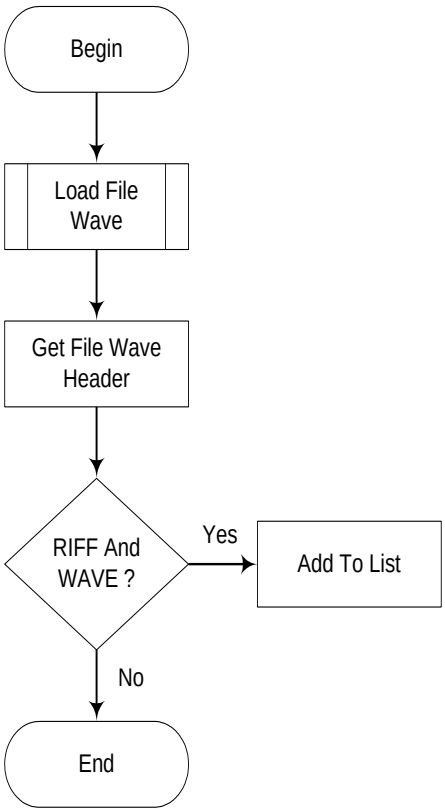
Pada bagian perancangan ini akan dijelaskan proses perancangan program beserta dengan perancangan *form* sebagai *user interface* dari program.

1. Perancangan Program

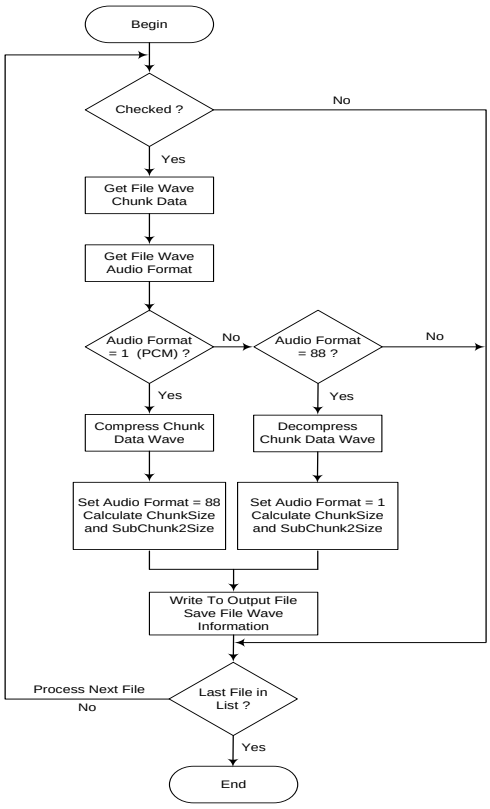
Algoritma atau *encoding* Huffman sebenarnya merupakan algoritma kompresi yang dapat diterapkan pada semua jenis baik untuk *file* biner maupun *file* teks. Algoritma ini efektif dengan rasio kompresi yang rendah jika terdapat banyak *redundancy data* atau perulangan data yang sama pada *file*.



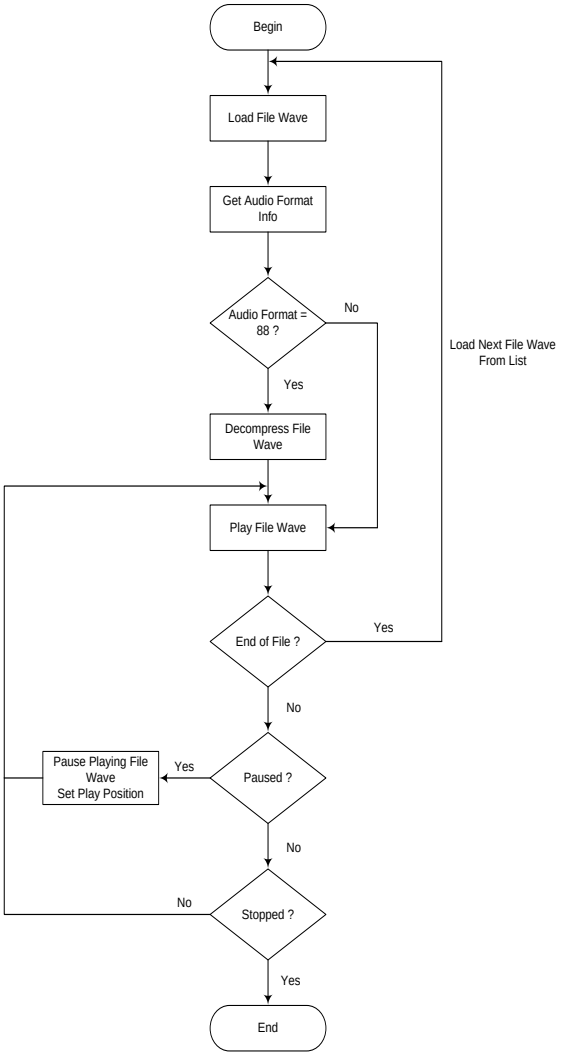
Gambar 5. Skema Struktur *Wave* Yang Diproses



Gambar 6. Diagram Alir Pembacaan File *Wave*



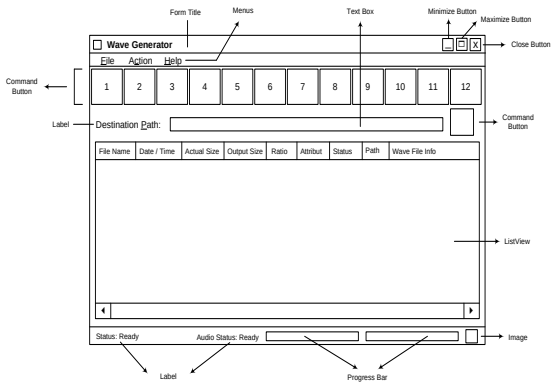
Gambar 7. Diagram Alir Proses Kompresi Dan Dekompresi File *Wave*



Gambar 8. Diagram Alir Memainkan File Wave

2. Perancangan Form

Berikut ini merupakan perancangan dari *form* utama program beserta dengan komponen Visual Basic yang dipakai.



Gambar 9. Rancangan *Form* Utama

IV. ALGORITMA DAN MPLEMENTASI

4.1 Algoritma

Pada bagian algoritma ini akan dijelaskan semua algoritma yang dipakai dalam proses perancangan program. Untuk keperluan ini perlu dideklarasikan terlebih dahulu tipe-tipe data dan konstanta seperti berikut ini yang mana akan dipakai sebagai jenis data dan kontanta pada bagian algoritma di bawahnya.

```
Type HUFFMANTREE
  ParentNode As Integer
  RightNode As Integer
  LeftNode As Integer
  Value As Integer
  Weight As Long
End Type
```

4.1.1 Algoritma Membentuk Pohon Huffman

```
Sub CreateTree(Nodes() As HUFFMANTREE, NodesCount As Long,
        Char As Long, Bytes As ByteArray)
        Dim a As Integer, NodeIndex As Long
```

4.1.2 Spesifikasi Perangkat Keras dan Perangkat Lunak

Program ini dijalankan dengan menggunakan perangkat keras (*hardware*) yang mempunyai spesifikasi minimal adalah sebagai berikut :

1. Prosesor Intel Pentium II 200 Mhz.
2. Memory 32 MB.
3. Harddisk 10 GB.
4. VGA *card* 1 MB.
5. Monitor dengan resolusi 800 × 600 *pixel*.
6. Keyboard dan Mouse

Adapun perangkat lunak (*software*) yang digunakan untuk menjalankan aplikasi ini adalah lingkungan sistem operasi MS-Windows 95/98 atau MS-Windows NT/2000/XP/Windows 7.

4.2 Cara Instalasi

Program ini tidak memerlukan cara instalasi yang khusus, dengan alasan bahwa semua *file* yang dibutuhkan oleh aplikasi ini dapat dikompilasi menjadi satu *file executeable*. Jadi untuk instalasi program ini cukup dengan meng-*copy file executeable*-nya (*Wave* Compressor.EXE) ke dalam lokasi *folder* yang dipilih pada *harddisk*. Jika program tidak dapat dijalankan lakukanlah instalasi dengan menjalankan *file* SETUP.EXE.

4.3 Pengujian Program

Tabel 1. Tabel Hasil Pengujian Proses Kompresi

No.	Nama File Wave	Ukuran File (byte)	Ukuran File Output (byte)	Rasio Kompresi	Lama Proses
1.	Windows XP Menu Command.wav	1.404	948	67,52 %	16
2.	Ringout.wav	5.212	4.129	79,22 %	31
3.	Windows XP Balloon.wav	6.400	4.819	75,30 %	62
4.	Ringin.wav	10.026	8.208	81, 87 %	47
5.	Windows XP Ringout.wav	22.070	20.129	91,21 %	125
6.	Windows XP Minimize.wav	22.580	17.262	76,45 %	128
7.	Windows XP Recycle.wav	22.816	18.378	80, 55%	132
8.	Windows XP Hardware Remove.wav	36.538	31.795	87,02 %	188
9.	Windows XP Hardware Fail.wav	36.614	32.769	89, 50 %	190
10.	Windows XP Hardware Insert.wav	36.636	30.756	83, 95 %	192
11.	Chimes.wav	55.776	48.627	87,18 %	266
12.	Ding.wav	80.856	70.725	87,47 %	406
13.	Windows XP Logoff Sound.wav	179.704	141.211	78,58 %	735
14.	Windows XP ShutDown.wav	282.608	247.861	87,70 %	1.203
15.	Windows XP Startup.wav	424.644	368.269	86.72 %	1.781

Dari hasil pengujian proses kompresi didapat bahwa rasio mempunyai *range* antara 67,52% untuk nilai terendah dan tertinggi 91,21%. Jika dicari hasil rasio kompresi tersebut secara rata-rata adalah sebesar 82,11%. Ini berarti ukuran *file* hasil adalah 0,8211 kali ukuran *file* semula dan pengurangan ukuran *file* sebesar (100% – 82,11%) = 17,89%. Nilai ini cukup terutama dalam mengkompresi *file Wave* berukuran besar misalnya berukuran di atas 10 MB.

V. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan pembahasan dari bab-bab sebelumnya yang telah dilakukan maka dapat diambil beberapa kesimpulan sebagai berikut:

1. Reduksi ukuran *file* yang diperoleh dengan algoritma Huffman ini berkisar dari *range* 20% hingga 40%. Jadi dapat dikatakan dengan rasio kompresi ini algoritma Huffman sudah dikatakan baik dalam hal mengkompresi *file* khususnya *file Wave*.

2. Tingkat kompresi dipengaruhi oleh banyaknya nada yang sama dalam *file* Wave.
3. Kecepatan proses tidak bergantung pada data yang diproses tetapi berbanding lurus dengan ukuran *file Wave*, artinya semakin besar ukuran *file Wave* yang diproses maka semakin lama waktu prosesnya.
4. Proses dekompresi lebih cepat dilakukan dibandingkan dengan proses kompresi karena pada proses dekompresi tidak dilakukan lagi proses pembentukan pohon Huffman dari data melainkan hanya langsung membaca dari tabel *code* pohon Huffman yang disimpan pada *file* sewaktu proses kompresi.
5. File Wave yang telah dikompresi bila dilakukan proses kompresi sekali lagi maka ukuran *file* akan bertambah besar sedikit karena algoritma Huffman merupakan *optimal compression* jadi *file* yang dilakukan kompresi sebanyak dua kali maka proses terakhir tidak akan mereduksi ukuran *file* lagi. Terjadi pertambahan *byte* pada proses kompresi kedua kalinya karena program menyimpan struktur pohon Huffman dari hasil kompresi pertama.
6. *File Wave* yang telah dikompresi tersebut hanya dapat dimainkan dari program ini.

5.2 Saran

Untuk pengembangan lebih lanjut program kompresi pada *file* Wave ini, maka dapat diberikan beberapa saran sebagai berikut:

1. Untuk meningkatkan rasio kompresi maka algoritma kompresi Huffman dapat digabungkan dengan rasio kompresi yang lain seperti LZW.
2. Untuk proses *play back file* Wave ditambahkan fasilitas yang lain seperti untuk *looping*, tombol *next*, dan tombol *previous*
3. Untuk memainkan *file* Wave yang telah dikompresi agar proses dekompresi lebih cepat maka dapat dilakukan dengan teknik *streaming* dimana *file* tidak perlu dikompresi sampai utuh di *memory* tetapi bagian *file* yang hanya sebagian didekompresi tersebut langsung dimainkan.

DAFTAR PUSTAKA

Basalamah, Affah, **Teknologi Multimedia MP3**, PT. Elex Media Komputindo, Jakarta, 2001.

Hadi R, **Pemrograman Windows API dengan Microsoft Visual Basic**, PT. Elex Media Komputindo, Jakarta, 2001.

Halvorson M, **Microsoft Visual Basic 6.0 Professional, Step by Step**, PT. Elex Media Komputindo, Jakarta, 2000.

Microsoft Developer Network (MSDN) Library Visual Studio 6.0, Microsoft Corporation, 1998.

Shannon, C. E., **A Mathematical Theory of Communication**, The Bell System Technical Journal, Vol. 27, pp. 379 − 423, 623 − 656, July, October, 1948.

http://www.stanford.edu/CCRMA/Courses/422/projects/WaveFormat/, 2002

http://www.replaygain.hydrogenaudio.org/file_format_wav.html, 2002

http://www.stanford.edu/~udara/SOCO/lossless/huffman/algorithm.htm, 2001

http://www.prepressure.com/techno/compression1.htm, 2002

http://www.data-compression.com/index.html, 2001

Saju, Vinil, "The Huffman Compression", http://www.howtodothing.com, 2003 Ang, Woi, "Data Structures and Algorithms: Huffman Encoding"

http://ciips.ee.uwa.edu.au/~morris/Year2/PLDS210/